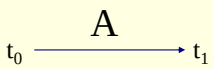
1 - Notion d'algorithme

Définitions

- Action : événement produit par un acteur, l'exécutant. Elle prend place pendant une période finie et produit un résultat bien défini et prévu.
- Indicateur : une action agit sur un ensemble d'indicateurs pouvant prendre des valeurs différentes. Un indicateur est caractérisé par son nom et sa valeur, il représente une information

Définitions

- État du système : l'ensemble des valeurs des indicateurs à l'instant t est appelé l'état du système à l'instant t.
- Effet d'une action : l'effet est caractérisé par 2 états du système :



t_0 = état initial de l'action A

t_1 = état final de l'action A

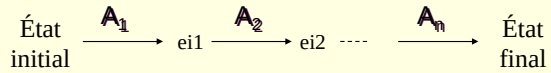
Définitions

- Spécification d'une action : c'est la description des états initial et final de l'action
- Algorithme : suite logique d'opérations permettant d'arriver à un résultat déterminé.

Un algorithme décrit un comportement en termes d'actions réalisables a priori, et d'informations identifiées dans le lexique de l'algorithme.

Définitions

- Construire un algorithme consiste à découvrir les actions qu'il faut organiser dans le temps, et à choisir la manière de les organiser pour obtenir le résultat escompté par leurs effets cumulés



Exemples d'algorithmes

- Une recette de cuisine
- Un mode d'emploi
- Une stratégie gagnante pour un jeu simple
- Une méthode systématique pour résoudre un problème mathématique, par exemple la résolution d'un système d'équations

Étapes de construction d'un algorithme

1) Préciser les conditions de travail du problème :
   - objets caractérisant les informations à manipuler
   - répertoire d'actions dont on dispose
2) Réfléchir à une méthode bien définie pour résoudre le problème.
3) Décrire cette méthode sans ambiguïté

Étapes de construction d'un algorithme

4) Évaluer la méthode décrite (temps d'exécution, nombre d'opérations, taille, etc..)
5) Réfléchir aux qualités et aux défauts de la méthode choisie
6) Réfléchir à d'autres méthodes et les comparer selon les critères de qualité définis

Jouons un peu…

Trouver une solution pour gagner au « jeu des plaquettes » dont voici la règle :

- Un meneur de jeu annonce un nombre entre 0 et 45.
- Les joueurs, au nombre de 5, doivent chacun choisir un nombre entre 0 et 9.
- Le total des chiffres choisis doit être égal au nombre annoncé par le meneur de jeu.
- Les joueurs n'ont pas le droit de se concerter pendant le choix du nombre qu'ils vont annoncer.

Jouons un peu…

- faire une partie pour comprendre le jeu (vous allez perdre !)
- réfléchir à une stratégie gagnante (vous ne devez plus perdre !)
- faire une partie pour vérifier que votre stratégie permet de gagner pour toutes les valeurs possibles

2 - Organisation séquentielle d'un calcul

2 - Organisation séquentielle d'un calcul

Exemple :

Écrire un algorithme qui, pour un nombre donné de secondes inférieur à un million, calcule son équivalent en nombre de jours, d'heures, de minutes et de secondes.

2.1 - Spécification, désignation, typage

Étapes de la construction d'un algorithme :

- Spécifier un problème consiste à expliciter les informations pertinentes pour une modélisation algorithmique, notamment les données et les résultats, puis à formuler les relations qui les caractérisent.
- Désigner les informations : c'est leur donner un nom

2.1 - Spécification, désignation, typage

Étapes de la construction d'un algorithme :

- Typer les informations, c'est donner leurs domaines de valeurs et recenser dans ces domaines les opérations pertinentes pour le problème.
- Formuler la solution algorithmique

Illustration sur l'exemple

Informations manipulées : le lexique de l'algorithme

lexique principal

ns : entier entre 0 et 999999 { donnée : nombre de secondes }
j : entier ≥ 0 { résultat : nombre de jours }
h : entier entre 0 et 23 { résultat : nombre d'heures }
m: entier entre 0 et 59 { résultats : nombre de minutes }
s : entier entre 0 et 59 { résultats : nombre de secondes }

Relation entre données et résultats :
ns = (86400 * j) + (3600 * h) + 60 * m + s

Illustration sur l'exemple

Algorithme principal
{ ns = n j = ? h = ? m = ? s = ? }
j ← ns div 86400
r1 ← ns mod 86400
*{ ns = n j = j_0 h = ? m = ? s = ? ns = j_0 * 86400 + r1}*
h ← r1 div 3600
r2 ← r1 mod 3600
{ ns = n j = j_0 h = h_0 m = ? s = ?
*ns = j_0 * 86400 + h_0 * 3600 + r2 }*
m ← r2 div 60
s ← r2 mod 60
{ ns = n j = j_0 h = h_0 m = m_0 s = s_0
*ns = j_0 * 86400 + h_0 * 3600 + m_0 * 60 + s_0 }*

Illustration sur l'exemple

lexique principal (modifié)
ns : entier entre 0 et 999999 { donnée : nombre de secondes }
j : entier ≥ 0 { résultat : nombre de jours }
h : entier entre 0 et 23 { résultat : nombre d'heures }
m: entier entre 0 et 59 { résultats : nombre de minutes }
s : entier entre 0 et 59 { résultats : nombre de secondes }

r1 : entier entre 0 et 86399 { intermédiaire : ns modulo 86400 }
r2 : entier entre 0 et 3599 { intermédiaire : r1 modulo 3600 }

2.2 - Lecture de données et écriture de résultats

On dispose de deux opérations :

- Lire : lecture de données du clavier **cl**
  Forme : **cl**.lire (liste des informations lues)
  Exemple : cl.lire(A,B)
- Écrire : affichage des résultats sur l'écran **e**
  Forme : **e**.écrire (liste des valeurs écrites)
  Exemple : e.écrire("Total TTC: ",TT * 1,196)

2.2 - Lecture de données et écriture de résultats

- On dispose de deux opérations :
- Lire : lecture de données du clavier **cl**
  Forme : **cl**.lire (liste des informations lues)
  Exemple : cl.lire(A,B)
  Définies dans le lexique
- Écrire : affichage des résultats sur l'écran **e**
  Forme : **e**.écrire (liste des valeurs écrites)
  Exemple : e.écrire("Total TTC: ",TT * 1,196)

2.2 - Lecture de données et écriture de résultats

- On dispose de deux opérations :
- Lire : lecture de données du clavier **cl**
  Forme : **cl**.lire (liste des informations lues)
  Exemple : cl.lire(A,B)
- Écrire : affichage des résultats sur l'écran **e**
  Forme : **e**.écrire (liste des valeurs écrites)
  - Constantes
  - Expressions
  - Noms d'informations

  Exemple : e.écrire("Total TTC: ",TT * 1,196)

Illustration sur l'exemple

lexique principal (complété):
cl : clavier { périphérique d'entrée }
e : écran { périphérique de sortie }
Algorithme principal
{ ns = ? j = ? h = ? m = ? s = ? }
cl.lire (ns) *{ ns = n }*
j ← ns div 86400 ; r1 ← ns mod 86400
*{ ns = n j = j_0 h = ? m = ? s = ? ns = j_0 * 86400 + r1}*
h ← r1 div 3600 ; r2 ← r1 mod 3600
*{ ns = n j = j_0 h = h_0 m = ? s = ? ns = j_0 * 86400 + h_0 * 3600 + r2 }*
m← r2 div 60 ; s ← r2 mod 60
{ ns = n j = j_0 h = h_0 m = m_0 s = s_0
*ns = j_0 * 86400 + h_0 * 3600 + m_0 * 60 + s_0 }*
e.écrire(ns," = ", j ," jours ,", h, " heures ,", m, " minutes ,", s,
" secondes")

Les types de base de la notation algorithmique

- Entiers : **Z**

constantes : 156 -12 1998
opérations : + - * / div mod
+ - * : Z X Z → Z
div mod : Z X Z* → Z
/ : Z X Z* → **R**
comparaisons : = ≠ < > ≤ ≥
Z X Z → **B** = { vrai, faux }

Les types de base de la notation algorithmique

- Réels : **R**

constantes : 3,14159 -0,001 1.2E5
opérations : + - * /
+ - * : **R** X **R** → **R**
/ : **R** X **R*** → **R**
comparaisons : = ≠ < > ≤ ≥
R X **R** → **B** = { vrai, faux }
fonctions prédéfinies : **pent** et **pdec**

Les types de base de la notation algorithmique

- Booléens : **B** = { vrai, faux }

constantes : vrai, faux
opérations : et ou (binaires) non (unaire)
et ou : **B** X **B** → **B**
non : **B** → **B**
comparaisons : = ≠
B X **B** → **B**

Rappel : Table de vérité

A	B	A et B	A ou B	non A	non B	non A et non B
V	V	V	V	F	F	F
V	F	F	V	F	V	F
F	V	F	V	V	F	F
F	F	F	F	V	V	V

non (A ou B) = non A et non B

non (A et B) = non A ou non B

Les types de base de la notation algorithmique

- Caractères : **C**

  constantes : 'A' 's' ' ' '='

  opérations : néant !

  comparaisons : = ≠

  C X **C** → **B**

Attention aux notations !

'X' représente le caractère X majuscule

X est le nom d'une information

Les types de base de la notation algorithmique

- Chaînes de caractères : **C***

  constantes : "A" "Bon" ""

  opérations : & **C*** X **C*** → **C***

  comparaisons : = ≠

  C* X **C*** → **B**

Attention aux notations !

"X" représente la chaîne de caractères X majuscule

X est le nom d'une information

'X' représente le caractère X majuscule

Fonctions et opérations prédéfinies sur les chaînes

- concaténation : **&**

  CH & "jour"

- ajout d'un caractère à gauche d'une chaîne: ∘

  'c' ∘ "abcd" = "cabcd"

  'c' ∘ "" = "c"

- ajout d'un caractère à droite d'une chaîne: ∘

  "abcd" ∘ 'c' = "abcdc"

Fonctions et opérations prédéfinies sur les chaînes

- Fonctions prédéfinies
  - SousChaine(ch,d,long) désigne la sous-chaîne de ch de longueur long et commençant avec le d^{ième} caractère
    SousChaine("hello",1,1)= "h"
  - Longueur(ch) désigne la longueur de la chaîne ch
    c'est-à-dire le nombre de caractères composant ch
    Longueur("hello") = 5
    Longueur("") = 0
- Désignation d'un caractère particulier de la chaîne:
  - Soit **ch** une chaîne valant "hello"
  - ch[1] désigne le 1er caractère de ch, c'est-à-dire 'h'

Fonctions et opérations prédéfinies sur les chaînes

- Désignation d'un caractère particulier de la chaîne:
  Soit **ch** une chaîne valant "hello" :
  ch[1] désigne le 1er caractère de ch, c'est-à-dire 'h'
  ch[i] désigne le i^{ème} caractère de ch, **1 ≤ i ≤ longueur(ch)**
  ch[longueur(ch)] désigne ?

  → Le dernier caractère de la chaîne ch
  Attention : ch est une chaîne non vide